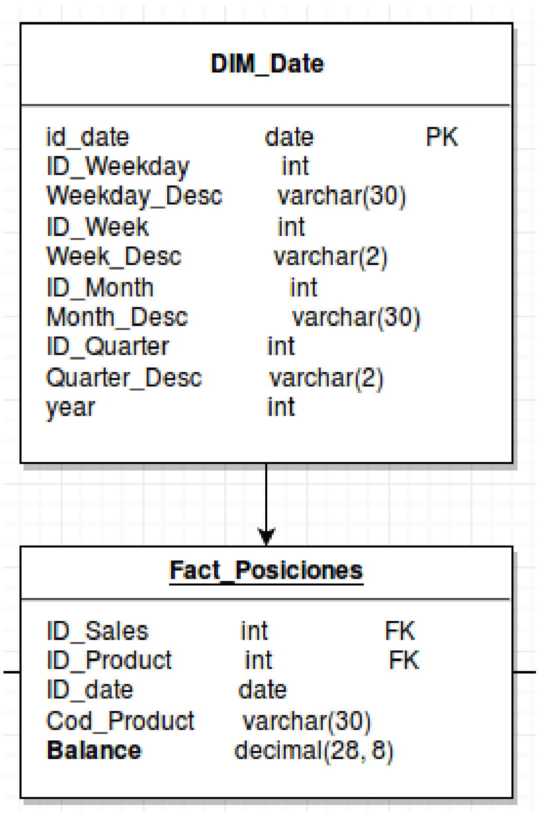
# Example of Tuning Cube I

**Date**: Dicember 2016

**Author**: Ramón Portolés, Alberto Linkedin a.ramonportoles@gmail.com

*Thanks to ShaoFeng Shi for help*

We can try to optimize a very simple Cube, with 1 Dim and 1 Fact table (Date Dimension)



**DIM_Date**

| | | |
|---|---|---|
| id_date | date | PK |
| ID_Weekday | int | |
| Weekday_Desc | varchar(30) | |
| ID_Week | int | |
| Week_Desc | varchar(2) | |
| ID_Month | int | |
| Month_Desc | varchar(30) | |
| ID_Quarter | int | |
| Quarter_Desc | varchar(2) | |
| year | int | |

**Fact_Posiciones**

| | | |
|---|---|---|
| ID_Sales | int | FK |
| ID_Product | int | FK |
| ID_date | date | |
| Cod_Product | varchar(30) | |
| **Balance** | decimal(28, 8) | |

Our Base line is:

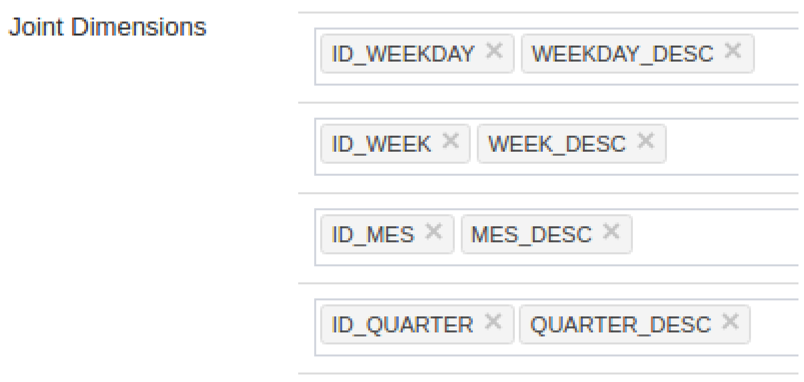
- One Measure: Balance, we calculate always Max, Min and Count
- All Dim_date (10 items) will be used as dimensions
- Input is a Hive CSV external table
- Output is a Cube in HBase with out compression

With this configuration, the results are: 13 min to build a cube of 20 Mb (Cube_01)

## Cube_02

Our first improve will be use Joint and hierarchy on Dimensions to reduce the carnality

We can put together all ID and Text of: Month, Week, Weekday and Quarter



Joint Dimensions

| |
|---|
| ID_WEEKDAY ✕ WEEKDAY_DESC ✕ |
| ID_WEEK ✕ WEEK_DESC ✕ |
| ID_MES ✕ MES_DESC ✕ |
| ID_QUARTER ✕ QUARTER_DESC ✕ |

Define Id_date and year like Hierarchy

The size down to 0.72 and time to 5 min

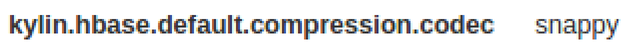
Kylin 2149, ideally, we can define define also these Hierarchies:

- Id_weekday > Id_date
- Id_Month > Id_date
- Id_Quarter > Id_date
- Id_week > Id_date

But for now, isn't possible use Joint and hierarchy together in one Dim :(

## Cube_03

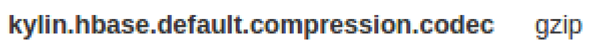
Now we can try compress HBase Cube with Snappy:



**kylin.hbase.default.compression.codec** snappy

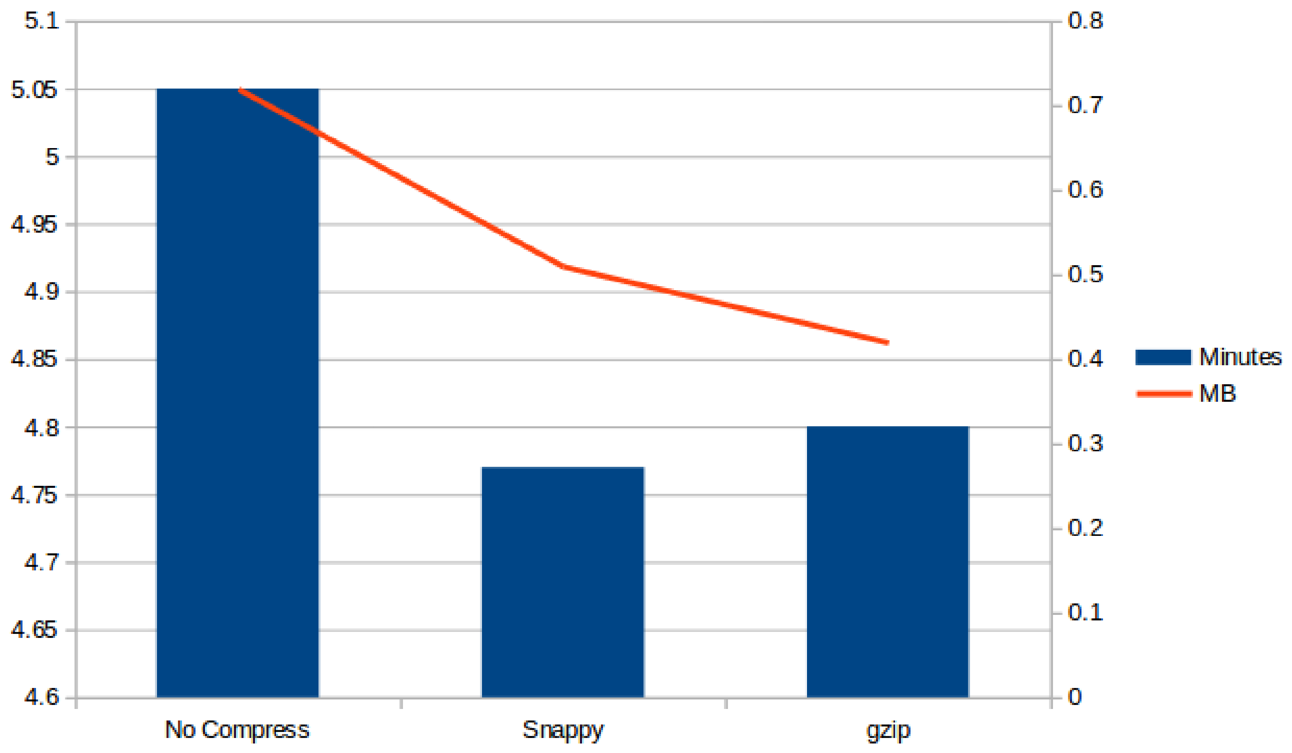
## Cube_04

Now we can try compress HBase Cube with gzip:



**kylin.hbase.default.compression.codec** gzip

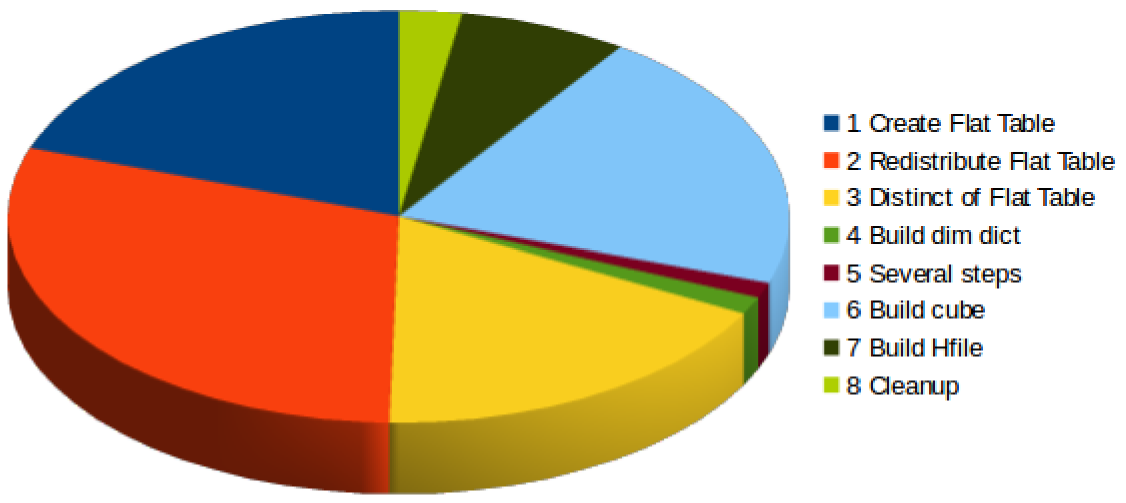
The results of compression output are:



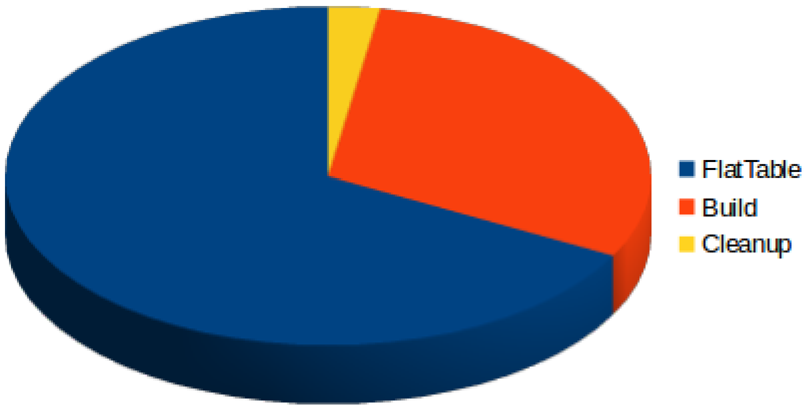
The difference between Snappy and gzip in time is very few 1% but in size is 18%

## Cube_05

Now the time distribution is like this:



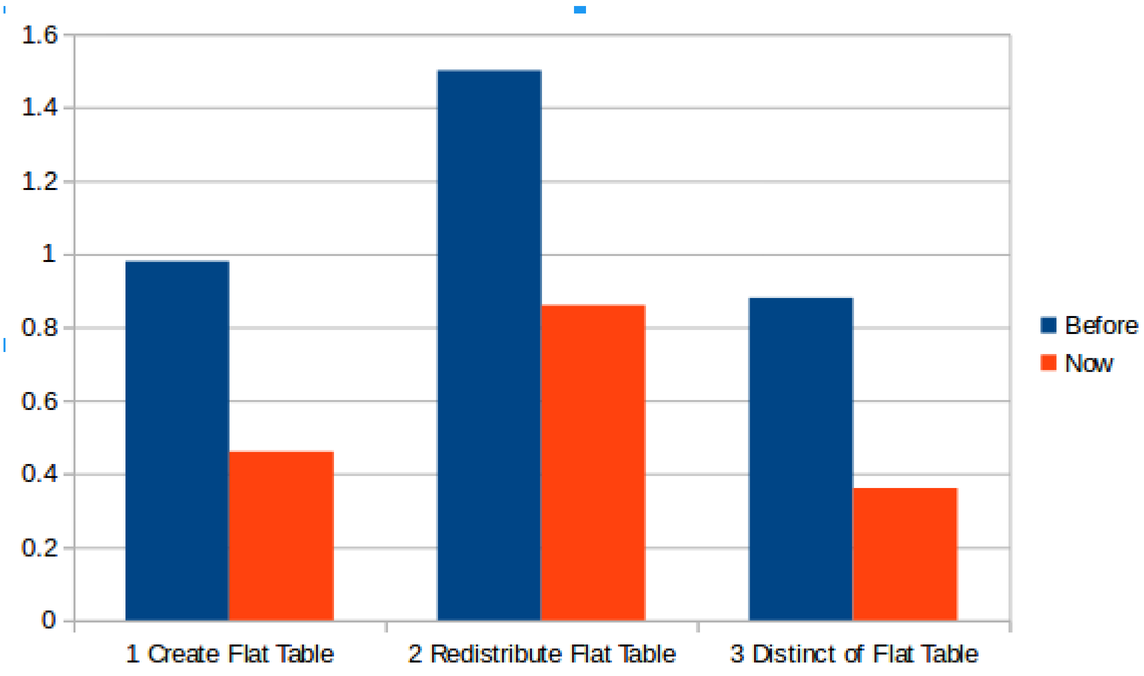
We can group detailed times in groups by concepts :



We can see the **67 %** is used to build / process flat table respect 30% to build cube

We are losing a lot of time in first steps !!

We can try to use ORC Format and compression on Hive (Snappy):



The time in three first steps (Flat Table) has been improved to the half :)

## Cube_06: Fail

If we see the distribution of rows

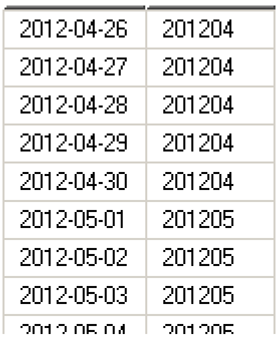
| | |
|---|---|
| Fact Table | 3.900.00 rows |
| Dim Date | 2.100 rows |

And see the query to build the flat table: (The idea is)

```
SELECT
        ,DIM_DATE.X
        ,DIM_DATE.y
        ,FACT_POSICIONES.BALANCE
FROM  FACT_POSICIONES  INNER JOIN DIM_DATE
        ON  ID_FECHA = .ID_FECHA
WHERE (ID_DATE >= '2016-12-08' AND ID_DATE < '2016-12-23')
```

The problem is, Hive in only using 1 Map to create Flat Table. Then lets go to change this undesirable behavior. Our solution is partition DIM and FACT by same columns

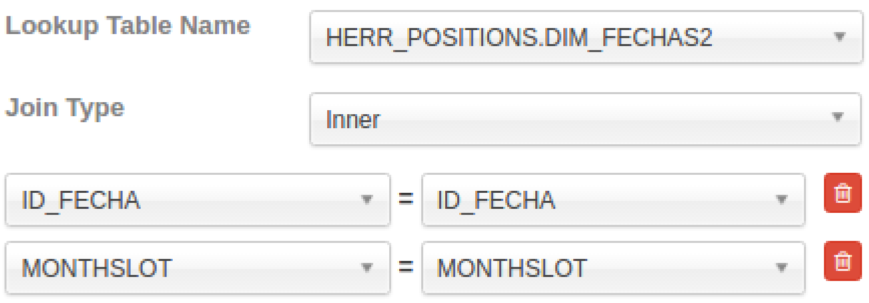
- Option 1: Use id_date as partition column on Hive table. This have a big problem: the Hive metastore is meant for few hundred of partitions not thousand (Hive 9452 there is an idea to solve this isn't in progress)
- Option 2:Generate a new column for this purpose like monthslot.



| | |
|---|---|
| 2012-04-26 | 201204 |
| 2012-04-27 | 201204 |
| 2012-04-28 | 201204 |
| 2012-04-29 | 201204 |
| 2012-04-30 | 201204 |
| 2012-05-01 | 201205 |
| 2012-05-02 | 201205 |
| 2012-05-03 | 201205 |
| 2012-05-04 | 201205 |

The same column will be add to dim and fact tables

Now the data model need be cached, add this new condition to join



Lookup Table Name: HERR_POSITIONS.DIM_FECHAS2

Join Type: Inner

ID_FECHA = ID_FECHA

MONTHSLOT = MONTHSLOT

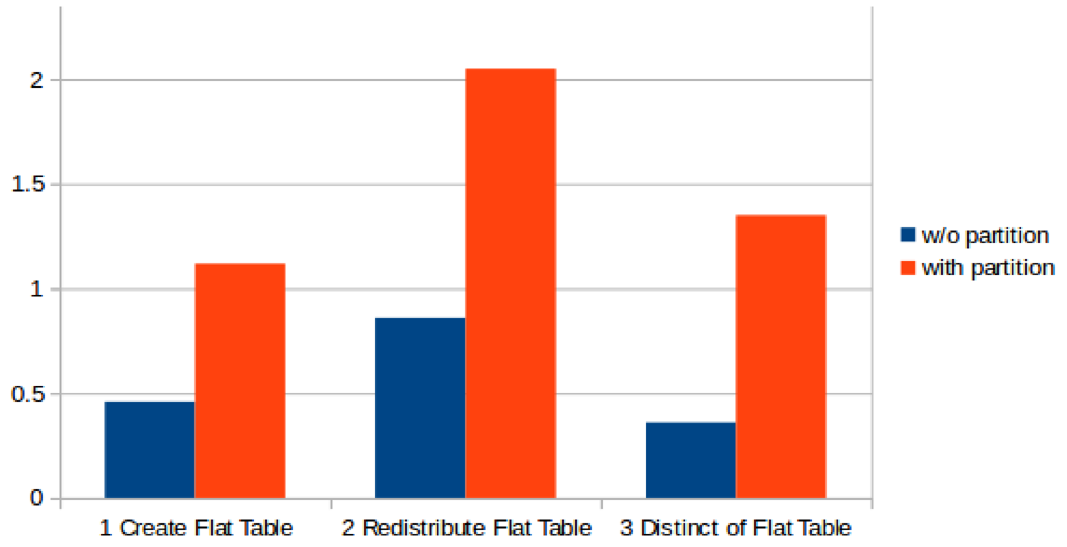
The new query to generate flat table will be similar to:

```
        SELECT
        FROM  FACT_POSICIONES  INNER JOIN DIM_DATE
```

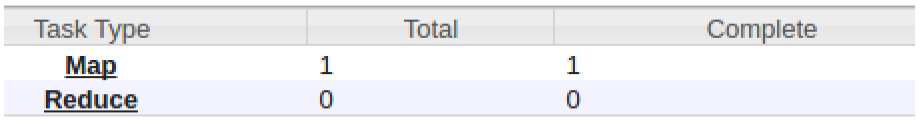
ON ID_FECHA = .ID_FECHA AND MONTHSLOT=MONTHSLOT

And launch the build of new cube with this data model

But The performance has worsened :( . I tried several test without solution

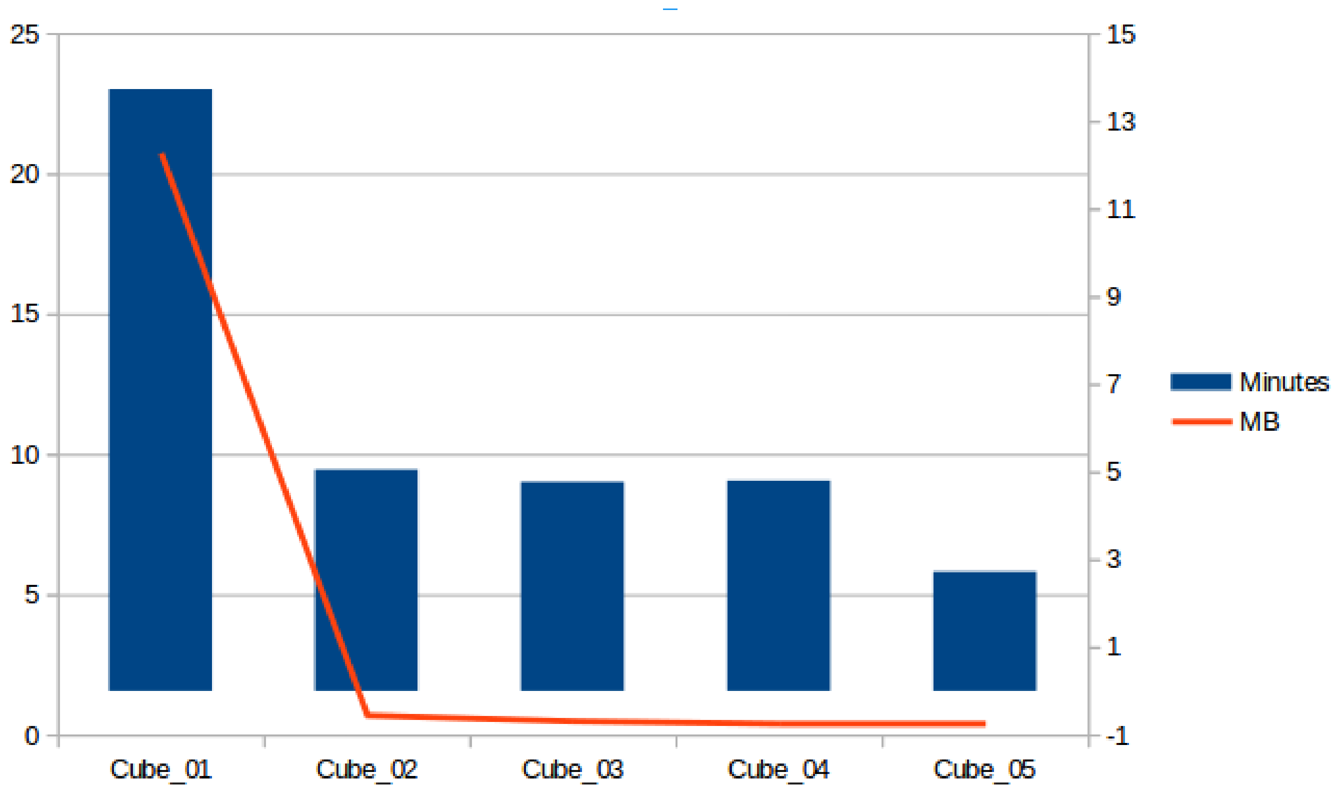


The problem is didn't use partitions to generate several Mappers



| Task Type | Total | Complete |
|---|---|---|
| Map | 1 | 1 |
| Reduce | 0 | 0 |

## The Final results


Cube_01
Cube_02
Cube_03
Cube_04
Cube_05
Minutes
MB